University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

EECS 122 I. Stoica
Fall 2008

FINAL EXAMINATION
December 18, 2008

INSTRUCTIONS—READ THEM NOW! This examination is OPEN BOOK/OPEN NOTES. There is no need for calculations, and so you will not require a calculator, Palm Pilot, laptop computer, or other calculation aid. Please put them away. All work should be done on the attached pages.

In general, if something is unclear, write down your assumptions as part of your answer. If your assumptions are reasonable, we will endeavor to grade the question based on them. If necessary, of course, you may raise your hand, and a TA or the instructor will come to you. Please try not to disturb the students taking the examination around you.

_____ SID: _____
(Signature)

_____ Discussion Section (Day/Time): _____
(Name—Please Print!)

QUESTION	POINTS ASSIGNED	POINTS OBTAINED
1	20	
2	18	
3	10	
4	12	
5	10	
6	15	
7	20	
8	5	
TOTAL	100 (+10)	

1. Miscellaneous (20 points)

a) *Worm propagation* (5 points). Each instance of the code-pink worm infects 1 machine in one second. If we start from a single infected machine, how many new machines get infected from second 10 to second 11? How many were infected by second 11?

From second 0 to 1, 1 machine gets infected. From second 1 to 2, 2 new machines get infected. From second N to N+1, 2^N new machines get infected. From second 10 to 11 we have 2^10 which is 1024 new machines. The total number of machines infected by second 11 is 2048.

Note: We also gave full credit for solutions which assumed that time starts at 1, instead of 0.

b) *Hidden terminal* (5 points). Is RTS/CTS guaranteed to eliminate hidden terminals? If yes, argue why; otherwise, present a counterexample.

No. For instance, we can have asymmetric links, and the sender does not hear a CTS from one receiver, but that receiver can hear what the sender is transmitting.

c) *Public key cryptography* (5 points). Assume you know the public key of entity X, but X has no information about you. Can you design a simple protocol to communicate *confidentially* with X in both directions (i.e., no one else knows what you and X are sending to each other)? If yes, specify the protocol, otherwise argue why this is not possible.

Yes. You send messages to X encrypted with X's public key and only X can decrypt them. Also, you send X in the first message your public key or a secret key with each X can encrypt the return communication. Note that this does not imply you are authenticated to X.

d) *Firewall* (5 points). You have to write the firewall rules for the following case. You are hosting a web service (http) on the machine with IP 10.10.10.10. You want to allow your web server to be accessed by anyone except the 20.20.10.0/24 network. You want to keep the traffic to a minimum. Write the rules using the form:

Accept/Deny	SrcAddr	DstAddr	SrcPort	DstPort
Deny	20.20.10.0/24	*	*	*
Accept	*	10.10.10.10	*	80
Accept	10.10.10.10	*	80	*
Deny	*	*	*	*

Note that for the outgoing communications we could allow for different than port 80 and give full credit.

2. Too many protocols? (18 points)

The most common link-layer protocol is Ethernet. On top of that runs the Network Layer protocol, IP. These two protocols have something in common: they have been remarkably successful in a huge range of different networks. Here's an idea: Maybe we don't need them both. Suppose that we eliminate IP entirely, and just used a single Ethernet network for the whole Internet. In particular:

- Instead of putting TCP or UDP packets inside IP packets inside Ethernet frames, we just put TCP or UDP packets directly in the Ethernet frames.
- Instead of identifying hosts with IP addresses, we use Ethernet addresses.
- Instead of routing with Link State, Distance Vector, or BGP, we use the spanning tree algorithm to find the paths packets will follow.

What could possibly go wrong with this plan? Say whether each of the following is **true or false**, and justify (no more than two-sentence).

(a) (3 points) There would not be enough Ethernet addresses to uniquely name all hosts. (If you agree, why does IP provide more addresses?)

False. Essentially every Internet host already has a unique Ethernet address today. Counting just the numbers, there are more ethernet addresses than IP addresses. The use of NATs effectively gives IP 2 bytes more space for addresses, but then it just matches Ethernet's 6 bytes. (And since port numbers are in UDP and TCP, not IP, it would be possible to use NATs directly with Ethernet.)

(b) (3 points) It would be harder for ISPs to control how data flows across their networks and which paths are used to deliver data. (If you agree, why do Link State, Distance Vector, or BGP give more control?)

True. Spanning tree gives you no control over the chosen tree. Link State and Distance Vector allow you to configure link weights, and BGP gives even more control over the particular route chosen for each destination. It would also be harder to control routes because Ethernet addresses have no relation to geographic location or ISP ownership.

(c) (3 points) It wouldn't scale -- that is, some resource in routers/switches would be over-utilized due to the large size of the Internet. (If you agree, what resource wouldn't there be enough of, and why wouldn't IP-level routing have this problem?)

True. Possible reasons: (1) Switches have to store one entry for every host on the network in their routing table. This is not as much a problem for IP, because hosts are grouped in blocks (prefixes) of IP addresses, and the router has only one entry for the whole prefix. (2) Switches initially flood messages throughout the entire network, until they learn hosts' locations. (3) Even after flooding, the root of the tree will be highly congested.

(d) (3 points) Efficiency would be worse: spanning tree would not send data along shortest paths, unlike Link State or Distance Vector.

True. Switches build a single tree of fewest-hop paths to the root, not a shortest path tree for every destination.

(e) (3 points) It would work fine for UDP, but not for TCP: Ethernet cannot support the guaranteed-delivery semantics and congestion control that TCP needs.

False. Both ethernet and IP provide the same reliability guarantees: none. TCP builds upon this to provide a reliable protocol, and you could run TCP over ethernet with minimal changes to the protocols.

(f) (3 points) Configuration would be much more difficult because the spanning tree algorithm has many parameters that need to be set correctly.
False. Actually, ethernet is plug-and-play and the other protocols are much more complicated.

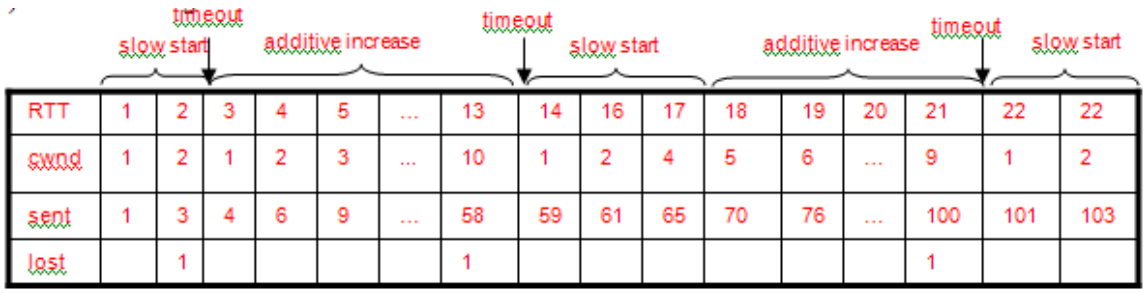
3. TCP (10 points)

A host uses TCP to send a 100KB file. Assume all the packets it sends have size 1000B (not including the IP and TCP headers). Thus, if there were no losses it would take exactly 100 packets to send the entire file. However, due to congestion, three packets are lost: the 2nd, the 50th, and the 98th packet sent by the sender. (Note that due to retransmission(s), the 50th sent packet is not the 50th chunk of the file.) No other packets are lost.

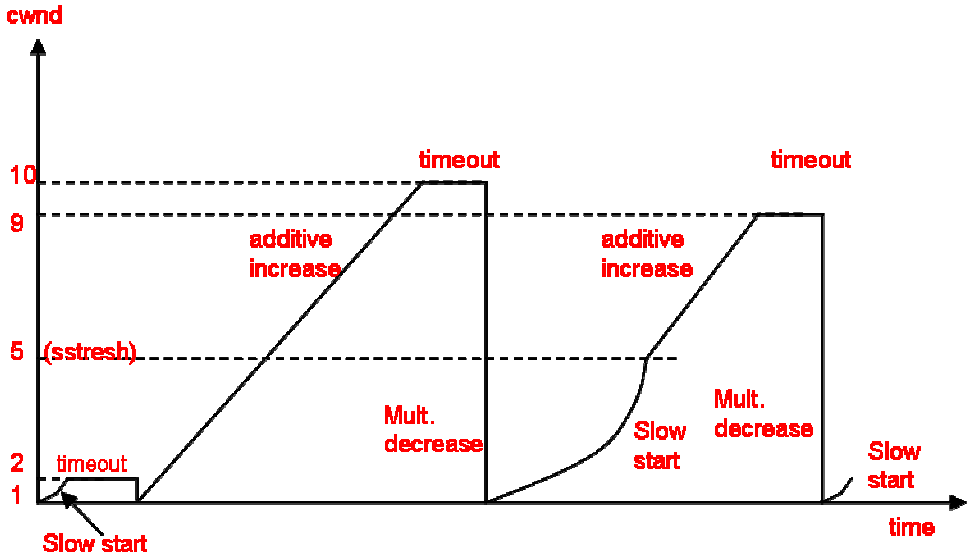
Plot the congestion window (cwnd) versus time for both TCP Tahoe and TCP Reno. Please make sure that you specify the size of cwnd when a packet loss occurs, size of ssthresh, and label the diagrams with slow start, AIMD, fast retransmit, and retransmission timeout where appropriate. Assume cwnd is measured in packets.

(a) (5 points) TCP Tahoe diagram:

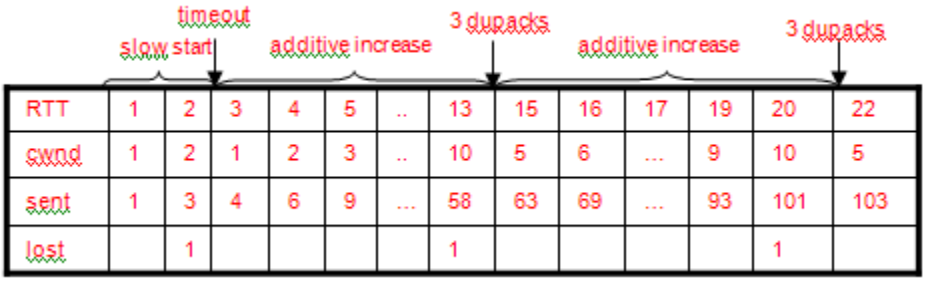
Note 1: We did not deducted points for minor mistakes such as saying that the 3rd timeout occurred at cwnd=10 instead of 9.



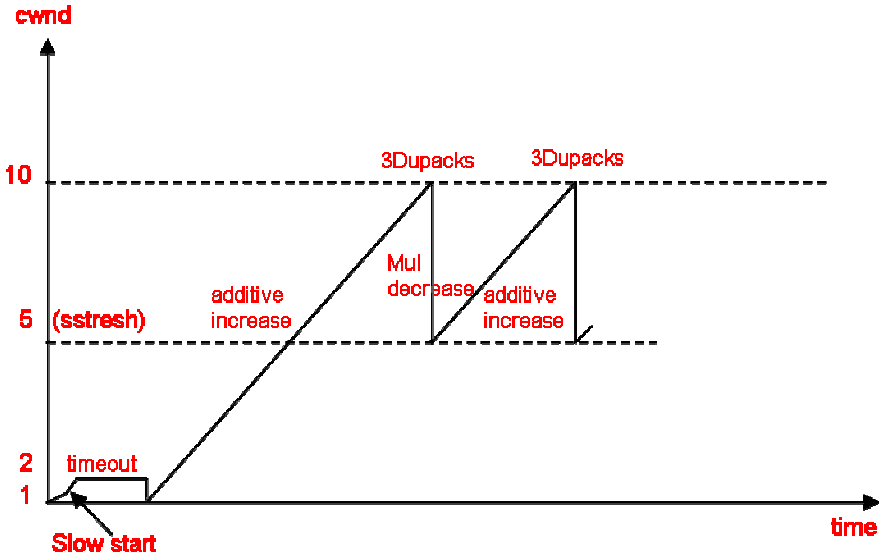
RTT	1	2	3	4	5	...	13	14	16	17	18	19	20	21	22	22
cwnd	1	2	1	2	3	...	10	1	2	4	5	6	...	9	1	2
sent	1	3	4	6	9	...	58	59	61	65	70	76	...	100	101	103
lost		1					1							1		



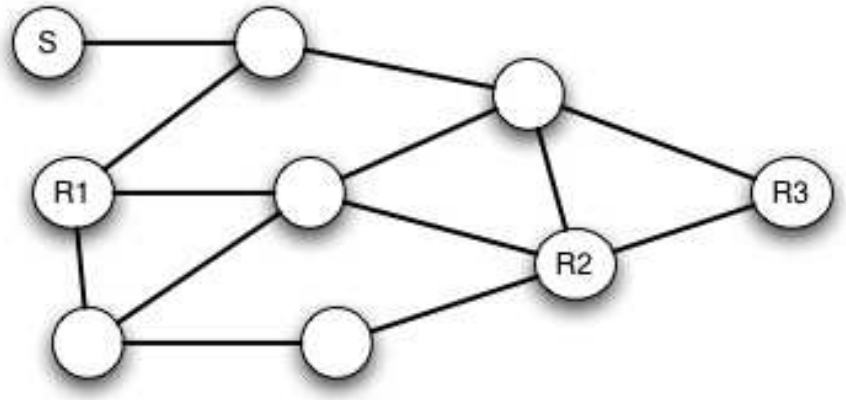
(b) (5 points) TCP Reno diagram:



RTT	1	2	3	4	5	..	13	15	16	17	19	20	22
cwnd	1	2	1	2	3	..	10	5	6	...	9	10	5
sent	1	3	4	6	9	...	58	63	69	...	93	101	103
lost		1					1					1	



4. Multicast (12 points)
Consider the following network of routers, where S is the sender of a packet and R1, R2 and R3 are interested in receiving that packet.



For each of the following questions, give the answer **and a brief (one-sentence) justification.**

First suppose S sends the data with a separate unicast to each of the receivers. Assume the network routes along shortest paths.

(a) (2 points) What is the *total number of packet forwards* that occur? (e.g., if two packets traveled across three links each, this number would be 6.)

8: S→R1: 2 forwards; S→R2: 3 forwards; S→R: 3 forwards

(b) (2 points) What is the *link stress?* (Recall that link stress is the maximum, over all links, of the number of times the data is sent across that particular link.)

3: there are copies of the packet sent on the link form S

Now instead of using unicast, we switch to a broadcasting algorithm, Reverse Path Broadcasting, that delivers packets to every node in the network. In this case:

(c) (2 points) What is the total number of packet forwards?

8: number of receivers

(d) (2 points) What is the link stress?

1: Reverse Path Broadcasting avoids sending duplicate packets.

Finally, instead of using broadcast, we switch to Truncated Reverse Path Broadcasting (i.e., the full Distance Vector Multicast Routing Protocol). Assume that pruning has already occurred, and the prune timers haven't expired. In this case:

(e) (2 points) What is the total number of packet forwards?

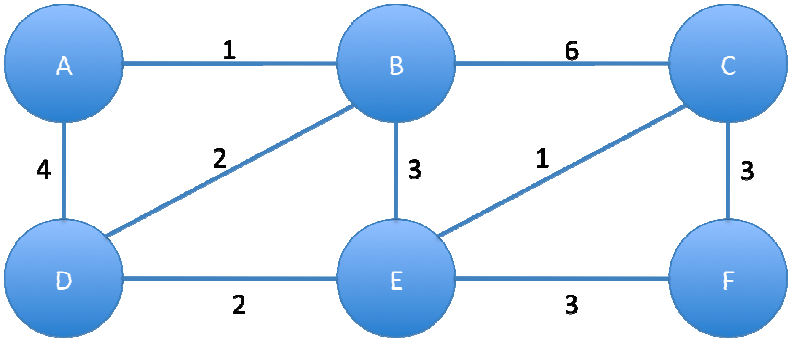
5: Truncated Reverse Path Broadcasting sends a copy along every link belonging to the tree routed at S and with leaves R1, R2, and R3.

(f) (2 points) What is the link stress?

1: Truncated Reverse Path Broadcasting avoids sending duplicate packets.

5. Routing (10 points)

a) (5 points) Router A is running a link state protocol and has received the topology shown below, on which it runs Dijkstra's algorithm to compute shortest paths. Fill in the table below showing the steps of Dijkstra's algorithm. In this table, the following notation is used: D(X) is the current distance from router A to router X, p(X) is the previous hop on the current path from A to X, and N' is the set of routers for which the shortest path has been determined.



step	N'	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	1,A	inf	4,A	inf	inf
1	AB		7,B	3,B	4,B	inf
2	ABD		7,B		4,B	inf
3	ABDE		5,E			7,E
4	ABDEC					7,E
5	ABDECF					

b) (5 points) Your friend tells you that he just discovered a huge improvement in link state routing. He plans to factor in congestion when performing routing. Explain the main challenge this would pose, and why.

You should ask him whether he has designed a mechanism to avoid oscillations in route selection. It is very challenging to avoid oscillations while being responsive to congestion.

6. Token bucket (15 points)

a) (5 points) Calculate the token bucket parameters rate r and bucket depth b for a flow with the following traffic requirements: the maximum rate is R = 20Mbps, the maximum rate can only be used up to 4 seconds, and in the first 10 seconds up to 140Mb can be transmitted.

The data sent in the first 4 seconds at the rate R is 80Mb. Then, in the next 6s another 60Mb need to be sent at the rate r. This means r = 10Mbps.

To compute the value for *b*, we have $b/(R-r) = 4s$ (i.e. *b* is depleted at the rate *R-r* up to its exhaustion), thus *b* = 40Mb.

b) (5 points) Router X wants the traffic of this flow to experience a delay of at most 1 second. In this case, what is the minimum rate r_a that X needs to allocate to the flow?

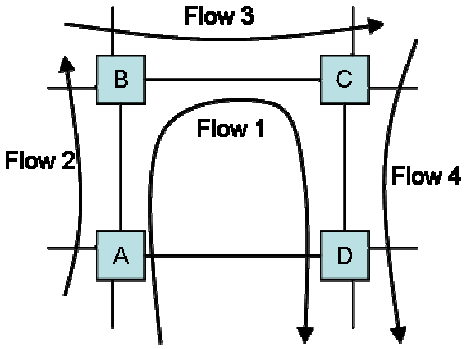
In the worst case, running at rate r_a, for time 4s+1s, we need to transmit the maximum received burst (80Mb), and thus, $r_a = 80/(4+1)$Mbps = 16Mbps.

c) (5 points) Given the requirement of part (b), what is the minimum buffer size that router X should assign for the above flow?

The maximum required buffer is given by the maximum burst size minus the transmitted size up to 4s (the knee), thus Ba = 80Mb – 16*4Mb = 16Mb.

7. Scheduling (20 points)

Consider the network below consisting of four routers. Every link has capacity of 1Mbs, and every flow sends data at 1Mbps. Assume that all flows are UDP and use the same packet size.



In words, Flow 1 uses links AB, BC, CD; Flow 2 uses link AB; Flow 3 uses link BC; and Flow 4 uses link CD.

a) (5 points) What is the throughput of each flow, if all routers implement FIFO scheduling? In this case, assume that in case of congestion each packet is dropped with the same probability.

Flow 1: 1/4Mbps, Flow 2: 1/2Mbps, Flow 3: 2/3Mbps, Flow 4: 3/4Mbps. Because each link is congested (the sum of the arrival rates of the flows at each link is greater than 1Mbps), ach flow gets a throughput proportional to its arrival rate. On link AB, the arrival rates of both Flow 1and Flow 2 is 1Mbps, so each flow gets 0.5Mbps. On link AB, the arrival rate of Flow 1 is 0.5Mbps while the arrival rate of Flow 3 is 1Mbps, so Flow 3 will get 2/3Mbps, while Flow 1 will get 1/3Mbps. Finally, on link CD, the arrival rate of Flow 1 is 1/3Mbps, while the arrival rate of Flow 4 is 1Mbps, so Flow 4 will get 3/4Mbps, while Flow 1 will get 1/4Mbps.

b) (5 points) What is the throughput of each flow if all routers implement Fair Queuing?

Every flow gets 1/2Mbps. On each link, with Fair Queuing each flow will get half of the link capacity, as the arrival rate of each flow is at least 0.5Mbps, which is the fair share on that link.

c) (5 points) What is the throughput of each flow if all routers implement Weighted Fair Queuing, and each Flow *i* has weight *i*?

Flow 1: 1/5Mbps, Flow 2: 2/3Mbps, Flow 3: 3/4Mbps, Flow 4: 4/5Mbps. On link AB, Flow 1 gets 1/3Mbps while Flow 2 gets 2/3Mbps; on link BC Flow 2 gets 1/4Mbps, while Flow 3 gets 3/4Mbps; on link CD Flow 1 gets 1/5Mbps, and Flow 4 gets 4/5Mbps.

d) (5 points) Answer question (c) if each flow implements congestion control, i.e., it adapts its transmission rate to the maximum rate that avoids packet loss (instead of sending at 1 Mbps constantly, as in the above parts).

Flow 1: 1/5Mbps, all the other flows: 4/5Mbps. In this case, Flow 1 will send no more than 1/5Mbps as this its lowest throughput along the path. As a result, the other flows will use the extra bandwidth, i.e., every other flow will get 4/5Mbps.

8. Doubling TCP throughput (5 points)

In the AIMD phase, the cwnd of TCP increases by **one** every round-trip-time if there are no losses, and decreases by **half** its current value if there is a loss. How would you modify the AIMD constants to roughly double the TCP throughput? Briefly explain your answer.

Assume the TCP flow traverses a congested link with capacity much higher than the flow throughput.

Basically we want to emulate the congestion control behavior of two TCP flows. Every round trip time the aggregate congestion window (cwnd) of both flows increases by 2 (one for each flow), and when there is congestion (i.e., loss) the aggregate cwnd decreases by half (the cwnd of each flow decreases by half). Thus, to double the throughput of TCP, its cwnd should increase by **2** every RTT, and decrease by **half** if there is loss.